(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 June 2002 (27.06.2002)

PCT

(10) International Publication Number
WO 02/50700 A2

(51) International Patent Classification⁷: G06F 15/78

(21) International Application Number: PCT/GB01/04685

(22) International Filing Date: 19 October 2001 (19.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
0030994.8 19 December 2000 (19.12.2000) GB

(71) Applicant *(for all designated States except US)*: PICOCHIP DESIGNS LIMITED [GB/GB]; 14 Sydney Gardens, Bath BA2 6BZ (GB).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: CLAYDON, Anthony, Peter, John [GB/GB]; 14 Sydney Gardens, Bath BA2 6BZ (GB).

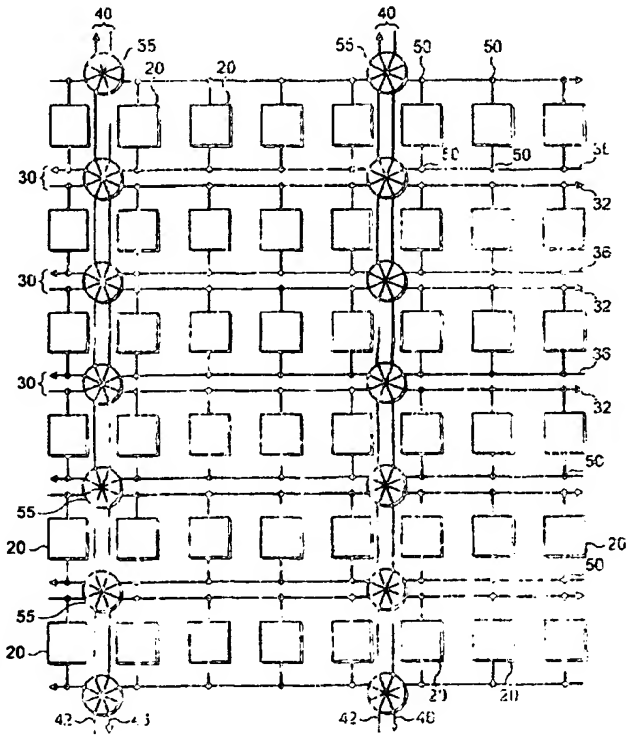
(74) Agent: O'CONNELL, David, Christopher; Haseltine Lake & Co., Imperial House, 15-19 Kingsway, London WC2B 6UD (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: PROCESSOR ARCHITECTURE



(57) Abstract: There is described a processor architecture having a plurality of processing elements, each element having at least one input port and at least one output port, each port having at least a data bus and a valid data signal line; and a bus structure which contains a plurality of switches which are arranged so as to allow an output port of any first processing element to be connected to the input port of any second processing element for a time interval, in which each processing element is enabled to set a value on the valid data signal line of its output port to a first logic state when the associated data bus contains a transfer value, and to a second logic state when the data bus does not contain a transfer value, and in which each processing element is further enabled to enter a waiting state for a predetermined time interval when the value on the valid data signal line of the associated input port is in the second logic state. This reduces the power consumption of the device.

IT, LU, MC, NL, PT, SE, TR), OAPI paten. (BF, B . CF, CG, CI, CM, GA. GN, GQ, GW, ML, MR. NE, SN, TD, TG).

Published:

-- *without international search report and to be republished upon receipt of that repor'*

For two-letter codes and other abbreviations, refer to the "Guidance Note on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

PROCESSOR ARCHITECTURE

This invention relates to a processor architecture, and in particular to an architecture which can be used in a wide range of devices, such as communications devices operating under different standards.

In the field of digital communications, there has been a trend to move as many functions as possible from the analogue domain into the digital domain. This has been driven by the benefits of increased reliability, ease of manufacture and better performance achievable from digital circuits, as well as the ever decreasing cost of CMOS integrated circuits. Today, the Analogue-Digital and Digital-Analogue Converters (ADC's and DAC's) have been pushed almost as near to the antenna as possible, with digital processing now accounting for parts of the Intermediate Frequency (IF) processing as well as baseband processing.

At the same time, there has been a vast improvement in the capability of microprocessors, and much of the processing for many narrowband communications systems is now performed in software, an example being the prevalence of software modems in PC's and consumer electronics equipment, partly because a general purpose processor with sufficient processing power is already present in the system. In the field of wireless communications there is extensive research in the field of software radio the physical layers of broadband communications systems require vast amounts of processing power, and the ability to implement a true software radio for third generation (3G) mobile communications, for example, is beyond the capability of today's DSP processors, even when they are dedicated to the task.

Despite this, there has never been a time when there has been more need for software radio. When second generation (2G) mobile phones were introduced, their operation was limited to a particular country or region. Also, the major market was business users and a premium could be commanded for handsets. Today, despite diverse 2G standards in the USA and different frequency bands, regional and international roaming is available and handset manufacturers are selling dual and triple band phones which are manufactured in their tens of millions. After years of attempts to make an international standard for 3G mobile the situation has now arisen where there are three different air interfaces, with the one due to replace GSM (UMTS) having both Frequency and Time Division Duplex (FDD and TDD) options. Additionally, particularly in the USA, 3G systems must be capable of supporting a number of legacy 2G systems.

Although a number of DSP processors are currently being developed that may be able to address the computational requirements of a 3G air interface, none of these show promise of being able to meet the requirements of a handset without the use of a number of hardware peripherals. The reasons for this are power and cost and size. All three are interrelated and controlled by the following factors:

1. The need for memory. Classical processor architectures require memory to store both the program and data which is being processed. Even in parallel Very Long Instruction Word (VLIW) or Single Instruction Multiple Data (SIMD) architectures, the entire processor is devoted to one task at a time (eg: a filter, FFT or Viterbi decoding), with memory required to hold intermediate results between the tasks. In addition, fast local instruction and data caches are required. Altogether, this increases the size and cost

of the solution, as well as dissipating power. In hard-wired architectures, data is usually transferred directly from one functional block to another, with each block performing DSP functions on the data as it passes through, thus minimising the amount of memory required.

2. Data bandwidth. In hard-wired solutions, all data is held locally, if necessary in small local RAM's within functional blocks. Some transceivers may contains several dozen small RAM's, and although the data bandwidth required by each RAM may be relatively small the overall data bandwidth can be vast. When the same functions are implemented in software running on a processor, the same global memories are used for all data and the required data bandwidth is enormous. Solutions to this problem usually involve the introduction of local memories in a multi-processor array, but the duplication of data on different processors and the task of transferring data between processors via Direct Memory Access (DMA) mean that the power dissipation is, if anything, increased, as is silicon area and consequently cost.

3. The need for raw processing power. In today's DSP processors, improvements in processing throughput are achieved by a combination of smaller manufacturing process geometries, pipelining and the addition of more execution units (e.g. arithmetic logic units and multiplier-accumulators). Improvements in manufacturing processes are open to all solutions, and so are not a particular advantage for conventional DSP processors. The other two methods both come with considerable overheads in increased area and power, not merely because of the extra hardware which provides the performance improvement, but because of the consequential increases in control complexity.

The processor architecture of the present

invention falls under the broad category of what are sometimes referred to as dataflow architectures, but with some key differences which address the needs of software. In fact, the invention provides a solution which is more akin to a hard-wired architecture than a DSP processor, with consequential size and power advantages. It consists of an array of processor and memory elements connected by switch matrices

According to the present invention, there is provided a processor architecture comprising

a plurality of processing elements, each element having at least one input port and at least one output port, each port having at least a data bus and a valid data signal line; and

a bus structure which contains a plurality of switches which are arranged so as to allow an output port of any first processing element to be connected to the input port of any second processing element for a time interval,

each processing element being enabled to set a value on the valid signal data signal line of its output port to a first logic state when the associated data bus contains a transfer value, and to a second logic state when the data bus does not contain a transfer value;

each processing element being further enabled to enter a waiting state for a predetermined time interval when the value on the valid signal data signal line of the associated input port is in the second logic state.

The waiting state is, for example, a low power sleep mode

This has the advantage that the power consumption of the device can be reduced when there is no data to be processed.

Preferably, the processing element is programmable in such a way as to set a predetermined time

internal

Preferably, the processing element is further enabled to load data from the data bus of its input port when the value of the valid signal data signal line of the associated input port is in the first logic state.

Preferably, the front port of each processing element is connected to the bus structure at a location in front of a location at which the corresponding output port is connected to the bus structure, in the direction of signal flow, such that, during a transfer time period the second processing element may set a second transfer value on the bus structure.

This achieves a further power saving in that, when data is not being transferred across a section of the bus structure, it does not need to be charged and discharged unnecessarily.

Preferably the processor elements include memory elements for storing received data, and/or processing elements including Arithmetic Logic Units and Multiplier Accumulators.

Preferably, each processing element has:

a first input for receiving data from a first bus;

a first output for transferring data to the first bus;

a second input for receiving data from a second bus; and

a second output for transferring data to the second bus.

The architecture of the preferred embodiment allows flexible data routing between array elements using a switch matrix. This means that the device is able to run many diverse algorithms required by a software radio concurrently, without having to reconfigure the array.

Reference will now be made, by way of example, to the accompanying drawings, in which:

Figure 1 is a schematic representation of a section of a processor, illustrating the basic architecture in accordance with the invention;

Figure 2 is an enlarged representation of a part of the architecture of Figure 1;

Figure 3 is an enlarged representation of another part of the architecture of Figure 1;

Figure 4 is an enlarged representation of another part of the architecture of Figure 1;

Figure 5 shows the illustration of elements in a typical array in accordance with the invention;

Figure 6 shows a first array element in the architecture of Figure 1;

Figure 7 shows a second array element in the architecture of Figure 1;

Figure 8 shows a first connection of the array element of Figure 7 in the array according to the invention;

Figure 9 shows a second connection of the array element of Figure 7 in the array according to the invention;

Figure 10 shows a third array element in the architecture of Figure 1;

Figure 11 shows a fourth array element in the architecture of Figure 1;

Figure 12 shows the format of data transferred between array elements; and

Figure 13 is a timing diagram illustrating the flow of data between array elements.

Figure 1 shows a part of the structure of a processor architecture 10. The device is made up of an array of elements 20 which are interconnected by buses and switches.

The architecture includes first bus pairs 30, shown running horizontally in Figure 1, each pair including a respective first bus 32 carrying data from left to right in Figure 1 and a respective second bus 36 carrying data from right to left.

The architecture also includes second bus pairs 40, shown running vertically in Figure 1, each pair including a respective third bus 42 shown carrying data upwards in Figure 1 and a respective fourth bus 46 shown carrying data downwards in Figure 1.

In Figure 1, each diamond connection 50 represents a switch, which connects an array element 20 to a respective bus 32, 36. The array further includes a switch matrix 55 at each intersection of a first and second bus pair 30, 40.

The data buses are described herein as 64-bit buses, but for some application areas it is likely that 32-bit buses will suffice. Each array element can be designed to be any one of the following:

an execution array element which contains an Arithmetic Logic Unit (ALU) or Multiplier Accumulator (MAC);

a memory array element, containing a RAM;

an interface array element which connects the processor to an external device, or

a switch control array element, which controls the operation of at least one switch matrix 55.

Each of these will be described in more detail below.

Figure 2 is an enlarged view of a part of the architecture of Figure 1, showing six array elements, 20A-20F. Each array element is connected to two 64-bit buses, 32, 36, which carry data in opposite directions. Apart from edge array elements (as shown in Figure 1), the horizontal buses are connected to two vertical buses, 42, 46, one carrying up and the other

down. The choice of bit width and vertical bus pitch is not fundamental to the architecture but these dimensions are presently preferred.

Each switch element 5 is a 2:1 multiplexer, controllable such that either of its two inputs can be made to appear on its output. Thus output data from an array element can be transferred onto a bus, and/or data already on the bus can be allowed to pass.

The switch matrix 5 includes four 4:1 multiplexers 501, 502, 503 and 504, which each controllable such that any one of their inputs can appear at their output.

The inputs of multiplexer 501 are connected to input connections 32a, 36a and 42a on buses 32, 36, 42 respectively, and to ground. The output of multiplexer 501 is connected to bus 42.

The inputs of multiplexer 502 are connected to input connections 32a, 36a and 46a on buses 32, 36, 46 respectively, and to ground. The output of multiplexer 502 is connected to bus 46.

The inputs of multiplexer 503 are connected to input connections 32a, 36a, 42a and 46a on buses 32, 36, 42 and 46 respectively. The output of multiplexer 503 is connected to bus 36.

The inputs of multiplexer 504 are connected to input connections 32a, 36a, 42a and 46a on buses 32, 36, 42 and 46 respectively. The output of multiplexer 504 is connected to bus 32.

Thus, in the switch matrix 5a, the input of any bus can be used as the source for data on the output of any bus, except that it is not possible to select the down-bus (i.e. the one extending from the top of the diagram in Figure 2, namely the fourth bus 46) as the source for the up bus (that is, the third bus 42), and, similarly, it is not possible to select the up bus (the third bus 42) as the source of the down bus (the fourth

bus 44.

[illegible] exceptions [illegible] which are not useful in practice. Internally, however, it is useful to have the left bus as a potential source for the right bus, and vice versa, for example when routing data from array element 20B to array element 20E.

As mentioned above, one of the inputs of each of the multiplexers 501, 502 is connected to ground. That is, each of the 64 bus lines is connected to the value 0. This is used as part of a power reduction method, which will be described further below.

Each of the multiplexers 501, 502, 503, 504 can be controlled by signals on two control lines. That is, a two-bit control signal can determine which of the four inputs to a multiplexer appears on its output.

Figure 5 is a view of the top-left hand corner of the array of Figure 1, showing the structure of a switch matrix 56 which is used when there is no input connection to a left-right bus 42 and of a switch matrix 57 which is used when there is no input connection to a left-right bus 42 or to a bus 46 running down.

The switch matrix 56 includes three 4:1 multiplexers 505, 506, 507, while the switch matrix 57 includes three 4:1 multiplexers 508, 509, 510. Compared to a switch matrix in the middle of the array, the number of input buses to multiplexers 505, 508 and 509 is reduced by one because there is no input bus entering from the left. Similarly, there is no input bus entering from the right as an input to multiplexer 510, but in this case the input bus which has been released is [illegible]. This is also the case for multiplexer 511, but in this case there is no input bus entering from the top of the switch matrix either, so the multiplexer has only two input buses.

Before the [illegible] of the array, no input buses

from the top or the left are available for multiplexer 506, which only has two inputs. Equivalent arrangements will be apparent for the bottom left, top-right and bottom-right corners of the array.

Figure 4 is a view of part of the top edge of the array of Figure 1, showing the structure of a switch matrix 58 which is used when there is no input connection to a bus 46 running down.

The switch matrix 58 includes two 4:1 multiplexers 511, 512. The inputs of available input buses to multiplexers 511 and 512 are connected by bus 46, but, in the case of multiplexer 511, one of the input buses has been replaced by the value zero. An equivalent structure for multiplexers on the bottom edge of the array is apparent.

Data transfer can be regarded as having three stages. Firstly, an array element puts the data on the appropriate output.

Secondly, multiplexers in the appropriate switch matrix or switch matrices are activated to make the necessary connections.

Thirdly, the destination array element loads the data.

Each of these events is controlled by a separate array element: the first and third by the source and destination array elements respectively, and the second by special switch control array elements. These are embedded into the array at regular intervals and are connected by control lines to all the multiplexers in the switch matrices which they control. Each array element controls the multiplexers immediately adjacent to its outputs, with the functions being performed separately on individual 16-bit fields. This allows several array elements in one column to put data on the same bus provided they are on different fields of the bus. This is particularly useful for those forms

such as Add-Compare-Select (ACS) in the Viterbi Algorithm, switching of the direction of horizontal and vertical buses is carried on the entire 64-bit bus and is controlled control signals.

Clearly, the three operations of source, switching and loading, although controlled independently need to be synchronised. This is achieved by restricting all data transfer operations to a series of predetermined cycles which are fixed at the time when the program is compiled and mapped onto the array. In a general purpose processor this restriction would be onerous, but it is actually helpful for many applications of the present invention.

As mentioned previously there are a number of types of array element, but they all must conform to three basic rules.

Firstly, they must have four input and output ports which connect to the abutting four sides of the array.

Secondly, they must run a program which is synchronised to the transfer cycles on the buses to which they are connected. In practice, this usually means that each array element must run a program loop which accesses the buses in a regular pattern which has a duration in clock cycles which is a power of two (e.g. 4, 8, 16 or 32 clock cycles).

Thirdly, they must obey instructions which appears on the buses during special control cycles, known as the array Control cycles.

A consequence of these rules is that, in the normal course of events, the entire program which an array element executes will be contained in the memory within the array itself. In fact more often than not, the program will be just one loop. It is possible to write programs which use instructions on this level to modify and overwrite and reload the instruction memory of the array element using the control cycles described above. An array

element has to [illegible] autonomously.

[illegible] say, [illegible] their programs when [illegible] arrives.

There are two types of [illegible]: Multiplier Accumulator (MAC) array elements and Arithmetic Logic Unit (ALU) array elements. These must be included in the [illegible] elements in appropriate [illegible] the target applications. For [illegible] applications [illegible] proportions, and Figure 1 shows an example of an array containing 256 array elements [illegible] for a communications transceiver. Figure [illegible] show the horizontal buses in the array and the positions of pairs of vertical buses 40 are shown as single lines.

As well as [illegible] Switch Control array elements, the example array of Figure 1 contains three interface array elements 80, 81 and 82. Array elements 80 and 81 are used for data input and output to the analogue portions of the transceiver and array element 82 is the interface to a microprocessor. Each of the four Switch Control array elements 83a to 83d controls the switch matrices in the corners of the array. For example, Switch Control array element 83a controls the switch matrices [illegible] the horizontal buses connected to the four [illegible] array elements 84.

Figure 2 shows the [illegible] of a Switch Control array [illegible] controller 94 and RAM 96. [illegible] with means of loading the RAM using the Array Control Protocol described below and [illegible] RAM. Data is loaded into the RAM from whichever the left bus 32 or right bus 34 [illegible]. Control [illegible] may

element is conventional, namely multiplexers 91 and 64-bit registers 93.

When the Switch Control element is set into its normal operating mode by means of Table signal 98, the address of RAM 95 is first set to zero and the first 160-bit word is read from and transferred to register 96. On each subsequent clock cycle, the RAM address is incremented and a new 160-bit word is loaded into register 96, until the address reaches 127 at which point it is reset to zero again and the process is repeated. The outputs of register 96 are applied directly to the select inputs of the multiplexers in the switch matrices 55 (Figures 3 and 5), so in this way all the switch matrices are controlled in a cyclical pattern lasting for 128 clock cycles. As previously noted, most areas of the array transfer data in cyclical patterns of a duration less than 128 clock cycles, but these are accommodated by repeating them within the 128 cycle pattern.

ALU and MAC array elements have the same interfaces to the array, differing only in the type of execution unit and associated instructions. Figure 7 shows an ALU array element, which will be used to describe these interfaces to the array.

Referring to Figure 7, three 64-bit registers, each shared from four 16-bit sub-registers 111a-121d, 121e-121h and 121i-121l can be connected to either of left bus 32 or right bus 36 through multiplexers 120, thus allowing them to be loaded from either bus. In response to instructions read from instruction store 123 and decoded in instruction decoder 113, any one 64-bit register can be connected to the left or right bus during one clock cycle and any combination of sub-registers loaded. For example, sub-registers 121a and 121b of 16-bit registers 121a-121l can be connected to the left bus to load data in bits

31:0 of left bus 34. Similar instructions cause data in the registers to be transferred over BUS 135 and stored back into the same or different registers 121, and still further instructions may modify the contents of these registers onto the left and right buses via multiplexer 124 and shifter 126. In the preferred embodiment, during the data cache cycle of a 64-bit register may be used to load data from an array bus, data from another may be used to send data to an array bus and ALU output can be used to modify the contents of registers, these tasks being accomplished by using separate fields [illegible] instructions.

Figure 8 shows the preferred [illegible] of a processor box 31 in Figure 7. RIGHT 131 and RIGHT 132 are 2-bit segments of a left bus 36 or a right bus 38. Control signals EN[3:0] 170 and SEL[3:0] 171 are controlled by instruction decode block 103 in Figure 7. Using these signals, each 16-bit multiplexer 170, 171 can be set to be equal to BUS 0, the right bus of the array element or zero.

Figure 9 illustrates how, in this case, the EDGE signal (described below) associated with the data on the bus can be allowed to pass along with the data or can be set by the array element.

Figure 10 shows the preferred embodiment of a Memory array element. This has many of the same features of the ALU array element described above, but in addition has RAMs 140, 141 and 142 via multiplexers 170, 171 and 172 which allow the [illegible] to R3 of 64-bit registers 120 to be used for data input to the RAMs, 16-bit address registers 141 are used for the address inputs to the RAMs and 16-bit auto-sequencer register 142 are used for [illegible]. Both address and data may be [illegible] under the control of [illegible] instructions [illegible] the

case of [illegible] the processes of loading data from the left or right buses 32 and 36 is also performed in exactly the same manner. The instructions stored in instruction store 140 and decoded in instruction decode unit 145 have an additional field included in the equivalent number of the PPU array elements. This additional field is used to control the reading of data from the bus and writing of data to them, these operations being performed in the same cycle as array move as and ALU operations.

Referring to figure 11 it can be seen that the addresses for the PPU may be calculated within the Memory Array Element being the uterine CPU and loaded in the sub-registers of the register file. Alternatively addresses may be provided on the array buses from another array element and used directly into registers.

In the example array of figure 5, memory array elements hold all the data which is processed by the execution array elements and there is an external global memory. However it will be clear that for a given application requiring a large amount of storage, access to external memory may be more easily appropriate. Furthermore, instructions which do not require the array elements to act as memory array elements but require the execution stores of the array elements may be loaded into the instruction stores of the array elements using the Array Control Unit.

Figure 12 shows how an individual Control Unit may be used with the processor architecture of the present invention.

Further it will be clear that only need to [illegible] the array elements for the

purp[illegible] ting the ADC [illegible] and to control [illegible] transfers sampled data onto the [illegible] element controller 152 can [illegible] instruction store and [illegible] Memory array elements [illegible] capable of being programmed to [illegible] analogue signal [illegible] register [illegible] at configurable points [illegible]

[illegible] are the Digital to Analogue [illegible] element, which [illegible] elements [illegible] latter [illegible] a general purpose [illegible] processing [illegible]

The [illegible] according to [illegible] described. However, [illegible] architecture comes [illegible] and in providing [illegible] common computation [illegible] physical layer [illegible] will [illegible] minimise [illegible] architecture [illegible] such as [illegible]

A number of [illegible] the [illegible] data [illegible] array [illegible] accordingly.

[illegible] there is valid [illegible] the

control of power consumption.

A major objective of the [illegible] is to keep the [illegible] array elements [illegible], eliminating the need for complex routing [illegible]. The Array Control Protocol (ACP) is used for the following:

[illegible] array elements when the array is [illegible]

Starting, stopping [illegible] array elements

[illegible] array elements during operation

Each array element has a Unique Identifier (UID), which is used to address [illegible] Array Control Words (ACWs) [illegible] information between array elements. When the [illegible] of a section of the bus is high [illegible] the data on the bus is an ACW. Figure [illegible] shows the structure of the [illegible] ACW.

When an ACW is put on the section of the bus to which an array element is connected, the array element must examine [illegible] in low-power sleep mode. If the [illegible] field of the ACW matches the UID of the array element or is equal to a designated broadcast address, the array element must interpret the FUNCTION field of the ACW and perform the required action. In the presently preferred embodiment of the invention, the following FUNCTION fields are defined:

Value	Function	Description
0	Reset	[illegible] internal [illegible]

1	Load Program 0	The DATA field contains a program word which must be placed in the first location in the program store of the array element
11	Load Program	The DATA field contains a program word which must be placed in the next location in the program store of the array element
100	Start	The array element must start executing program in program store
101	Stop	The array element must stop executing program in program store
110	Test	Not used in the
111	Dump	Place data from next location in the program store on the bus

An array may be comprised by an array element, but the array will normally include an element which is defined as the master array element and the master controller will generally [illegible]. The major function of the Array Control [illegible] is to load the program stores of the array elements with the code to be executed. Therefore, a host [illegible] array element which loads the program supplied by the user processor. This most likely will be the source of ACKs.

Unlike most processors, which are instruction driven, the processors of the present invention and its component array elements are data driven. That is, instead of [illegible] fetching an instruction, array elements execute instructions as a result of receiving data.

Once a program has been loaded into an array element and it has been started using the [illegible]RT Array Control Word it will begin to execute its instruction sequence. Each instruction which requires

it to [illegible] data when [illegible] data is present on the bus (signified by the [illegible] signal [illegible] being low) it must stop and wait until data is available. During the time it is stopped it goes into a low power sleep mode. When [illegible] each array element will examine the bus to see if data is specifically addressed to it by a field in the local interface which is polled to check if the data has arrived.

In an example where the demodulator in a demodulator using the architecture described herein, the demodulator will receive ADC input samples at a fixed rate which generally will be somewhat above the actual symbol rate. The front end of the demodulator will contain an interpolator, which uses the incoming data flow to phase lock the local analogue VCO to synchronise the sampling instants to the data, but the resampled data will be independent with respect to the processor system clock and data transfer sequences, meaning that the data could have been expected. (In fact the ADC sample clock need not be synchronised to the processor system clock at all, with synchronisation to the system clock being performed in the ADC interface array element). Using the data driven processor architecture of the present invention, where there is a "gap" in the incoming data, the array elements which are affected simply go to sleep" until data is available.

It should be noted that because all data transfers are synchronised to sequences which are defined by the time line controller and mapped to the processor, every element will be set for at least one of the sequences to which it is synchronised.

[illegible] in the [illegible] timing diagram, will be related to the [illegible] units (A and B) are [illegible] the [illegible] Successive

transfer [illegible] (LAST PEA SEQ). In the [illegible] on the second (as shown in the [illegible]) the [illegible] load being shown [illegible] LOST REQ [illegible] REQ [illegible] the BDVAL signals associated with the [illegible] loaded by array elements [illegible] sequence that, where no data is available [illegible] the BDVAL signal [illegible] in which [illegible] sequence in which [illegible] for array element B, the [illegible] until the [illegible] data is available for [illegible] of the array element does not affect [illegible] the subject.

Clearly, if [illegible] receive any data, there will be a [illegible] does not receive data [illegible] the array. However, the appropriate [illegible] particular point in the algorithm until [illegible] the time the program is written, some useful [illegible] which tend to occur naturally at points in the algorithm where data needs to be moved, [illegible] where a block of data has to be accumulated before it is processed means that the entire array [illegible] gaps which occur at the front [illegible]

In some cases, [illegible] array element does not receive [illegible] elements must be [illegible] multiple [illegible] from a memory [illegible] array, if the data [illegible] arrive [illegible] array element [illegible] prevented from sending [illegible] data past the [illegible] array element [illegible],

[illegible] sleep mode.

[illegible] BDVAL signal is [illegible] to it is [illegible].

[illegible]. Every [illegible] ion, which [illegible], it [illegible] signals (Ref. [illegible] Fig. 9) [illegible] such that BDVAL [illegible] one clock cycle, [illegible] and SEL[illegible] BUSCLK[illegible] to [illegible]. For example, if data is to be transferred [illegible] bus, then all of EN[illegible] are set to 1. To [illegible] transferred on bits [31:0] of the [illegible] and [0] are set to 1, the [illegible] are set to 1 if [illegible] the other bits of the [illegible] clock cycle. Figure [illegible] example of multiple array elements [illegible] to transfer data in [illegible] clock cycle, referring to Figure 2 where [illegible], it can be seen that array element [illegible] transfers data bits [31:0] of bus 18, whilst array element 200 transfers data on bits [illegible], [illegible] array elements [illegible].

During the clock [illegible] the Switch [illegible] switch [illegible] in switch [illegible] that [illegible] are routed to [illegible]. Reference is made to Figure 9 [illegible] destination array [illegible].

instruction which causes multiplexer 22 to select the bus on the input of the multiplexer 22, which is located at the end of the bus to which the BDVAL signal is applied. If the bus is not loaded, data is placed onto the array element after a number of clock cycles specified by parameter in the WAIT instruction field. During the time that the array element is waiting, the only activity that may occur on the array element is the execution of a port write which loads the wait period into a counter and resets it. When the count reaches zero, the array element will no longer examine the BDVAL signal and execution proceeds from the next instruction. Because the circuitry on the array element is very small compared to the rest of the array element, very little power is consumed while an array element is waiting.

As will be described in detail above, all array elements which are to be synchronized for data transfer also have a WAIT instruction. This instruction causes the array element to examine the BDVAL signal on the bus to its right but is ended early if the specified number of clock cycles if selected BDVAL signal is 0 before the data is loaded.

Throughout the above description reference has been made to methods of selecting various components in the array. There may be many possible methods, only one detail.

In order to minimize power consumption during data transfer over the array, it is important that wires and circuit segments are activated only when necessary. In order to minimize power, the default state of all bus lines is set to the [illegible]. Switch instructions are used to select the [illegible] that are being

used [illegible] 502 in Figures 5 and 6 [illegible] the edges and [illegible] Figures 3 and 4.

Not data [illegible] bus often not all 64 bits [illegible]. [illegible] as [illegible] is loaded [illegible] 0. If [illegible] bus [illegible] would have been [illegible], so the [illegible]

[illegible] if data is being transferred [illegible] array element 20E to array element 20F, [illegible] along right bus 30 which is [illegible] 20E, part [illegible] and beyond thus preventing [illegible] further segments of [illegible] occurring. All [illegible] for [illegible] switch boxes 51 to be set [illegible] is set to 0 [illegible] achieved by setting [illegible] to 0 and signals [illegible]. A field is provided [illegible] executed on an array element [illegible] is allowed to propagate [illegible] is stopped as just described. This [illegible] array elements to load the same data [illegible] the bus which are [illegible] clock cycle.

[illegible] architecture which [illegible] a rectangular [illegible] of terms of [illegible]

CLAIMS

1. A processor [illegible] comprising:

a plurality of [illegible] elements, each element having at least one input port and at least one output port, each port having a data and a valid data signal line; a

a [illegible] switches which connect a plurality of switches which are programmable to be either an output port of any first processing element to be connected to the input port of any [illegible] processing element for a time interval;

each processing element being enabled to see a value on the valid data signal line of the input port to a fixed logic state, the associated data being consumed, transferred, and the second logic state when the first has been consumed at a transfer rate;

each processing element being further enabled to enter a waiting state for a predetermined time interval when the value on the input port is not the second logic state associated input port is in the second logic state.

2. A processor architecture as claimed in claim 1, wherein the processing elements are programmable in such a way as to set the predetermined time interval.

3. A processor architecture as claimed in claim 1, wherein the processing elements are enabled to enter the waiting state for a predetermined time interval, to resume [illegible] signal line.

4. A processor [illegible] as claimed in claim 1, 2 or 3, where [illegible] enabled to read data from one of the input port when the valid [illegible] the [illegible] state.

5. A processor [illegible] as claimed in any preceding claim [illegible] input and output ports [illegible]

comprising a first number of bits, and wherein the transfer value [illegible] is less than or equal to the first number.

6. A processor architecture as claimed in claim 5, wherein the second processing element is enabled to load any number of bits less than or equal to the first number of bits.

7. A processor architecture as claimed in any preceding claim, wherein [illegible] processing element is connected to the bus structure at a location in front of a location at which the corresponding output port is connected to the bus structure, in the direction of signal flow, such that, during a transfer time period, the second processing element may set a second transfer value on the bus structure.

8. A processor architecture as claimed in claim 7, wherein [illegible] value set on the bus structure [illegible] processing element is enabled to set a predetermined value on the bus structure.

9. A processor architecture as claimed in any preceding claim, wherein the processing elements include memory elements, for storing received data.

10. A processor architecture as claimed in claim 9, wherein the processing elements include processing elements for processing received data, and memory elements for storing received data.

11. A processor architecture as claimed in any preceding claim, wherein the processing elements include Arithmetic Logic Units.

12. A processor architecture as claimed in any preceding claim, wherein the processing elements include multiplier elements.

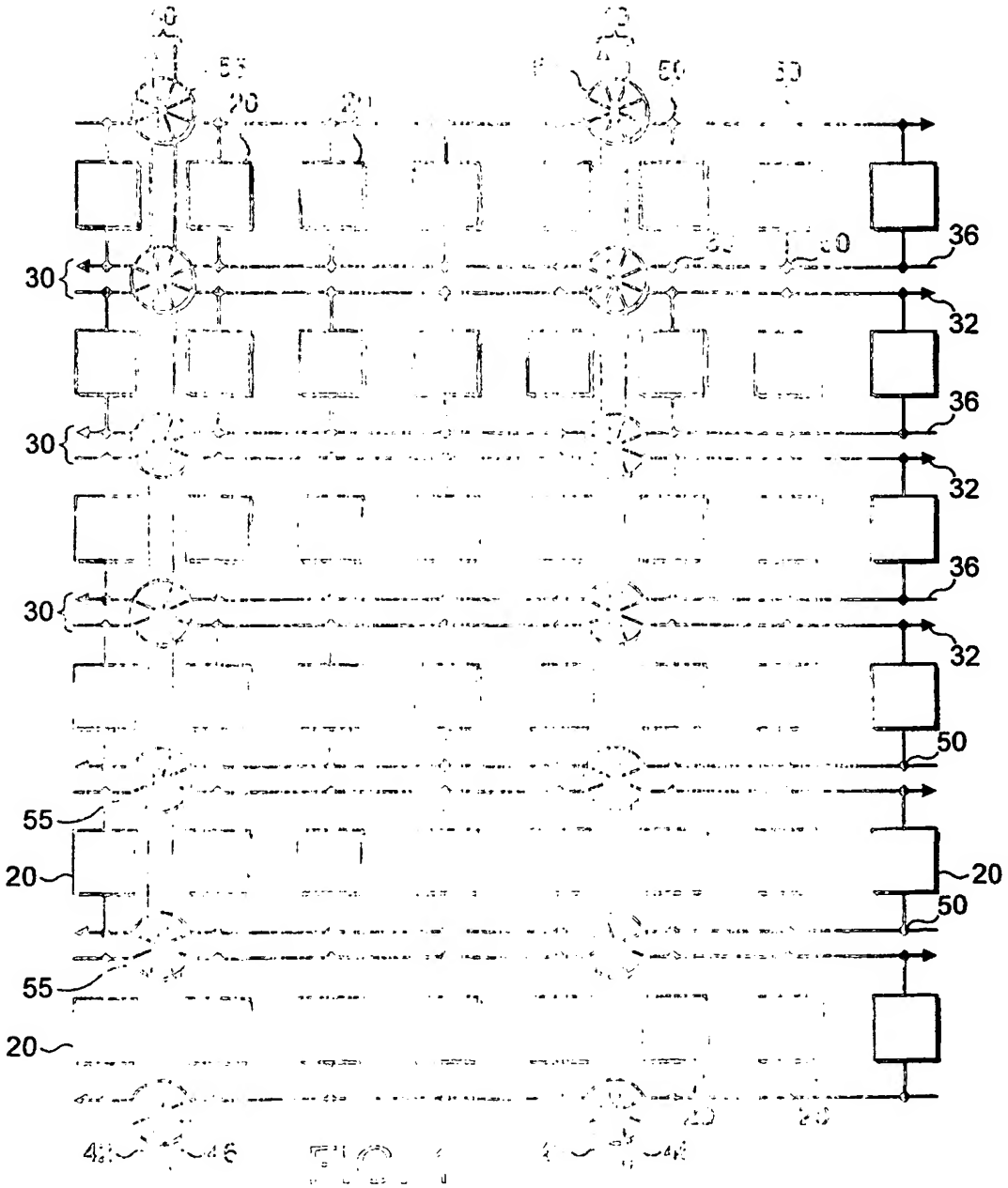
13. A processor architecture as claimed in any preceding claim, [illegible]

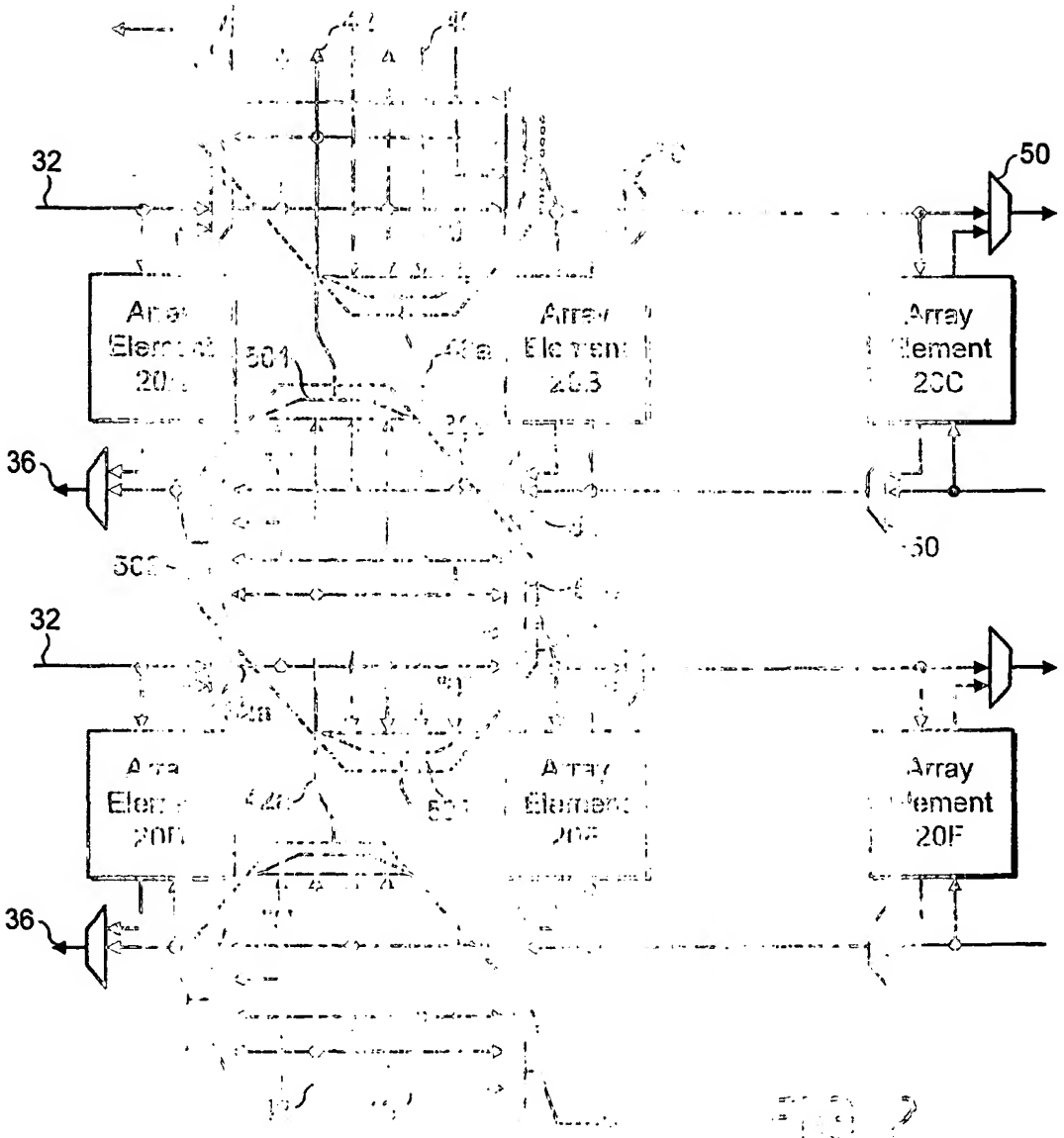
[illegible] first

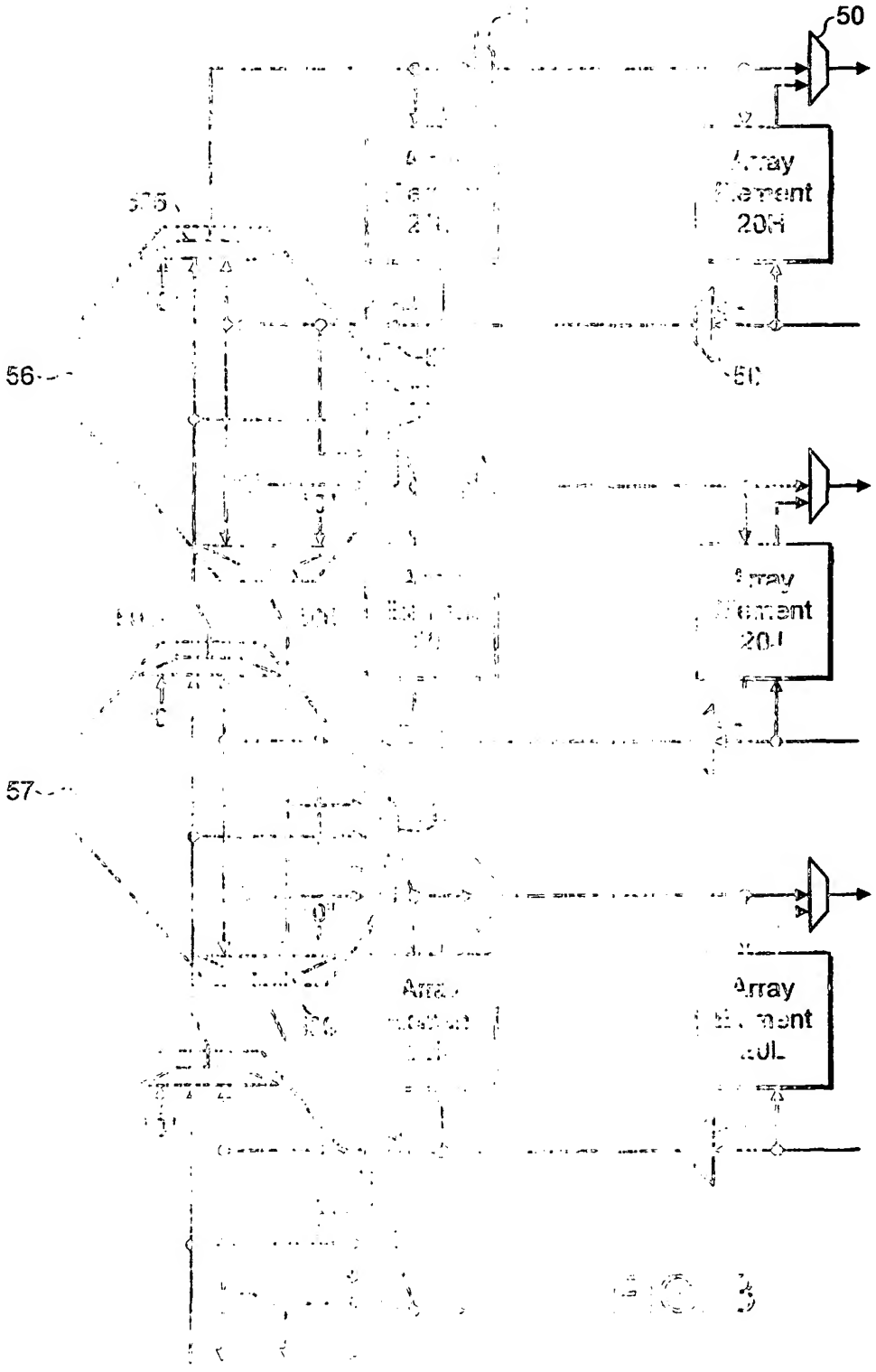
[illegible] the

[illegible]

[illegible] to the




32
50
Array Element 20C
36
32
Array Element 20F
36


50
Array
Element
20H
56
Array
Element
20I
57
Array
Element
20L

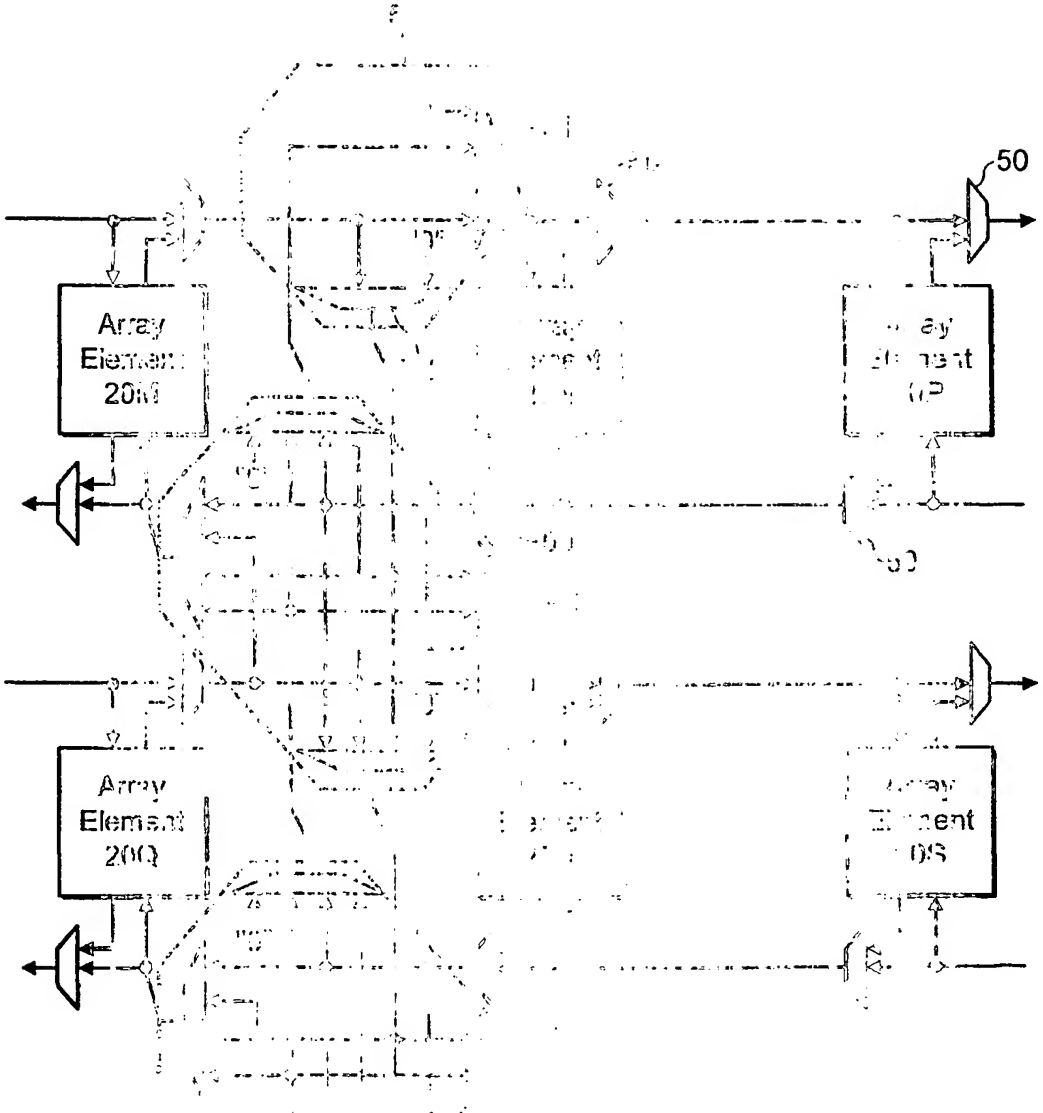


FIG. 4

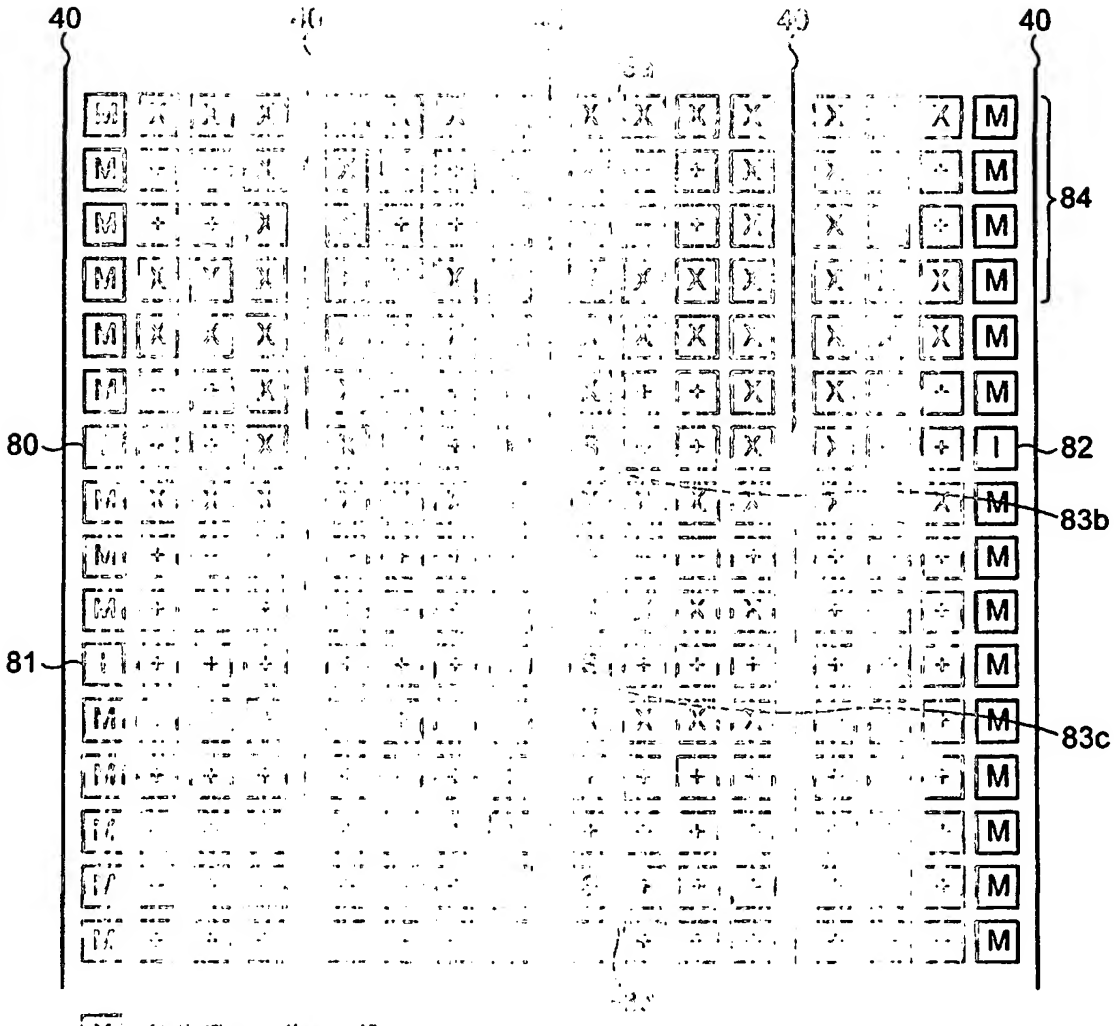
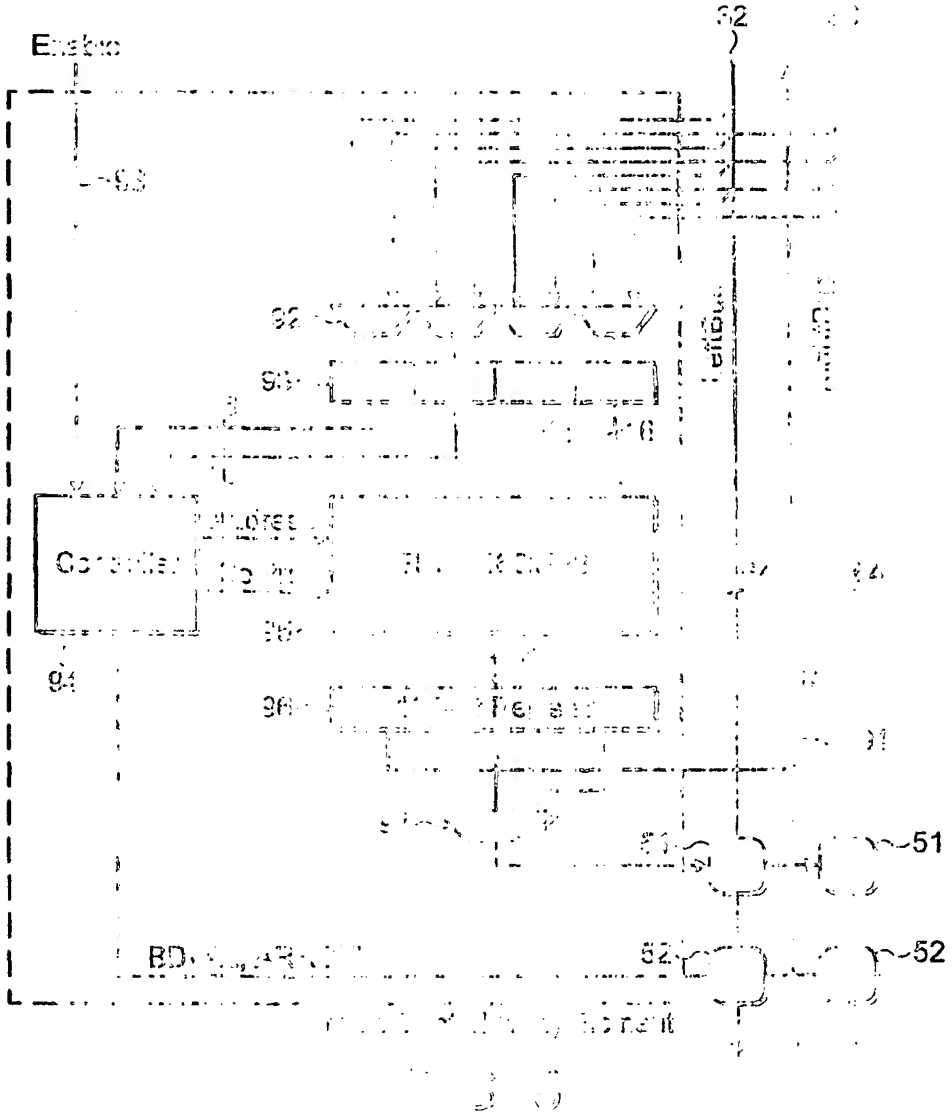
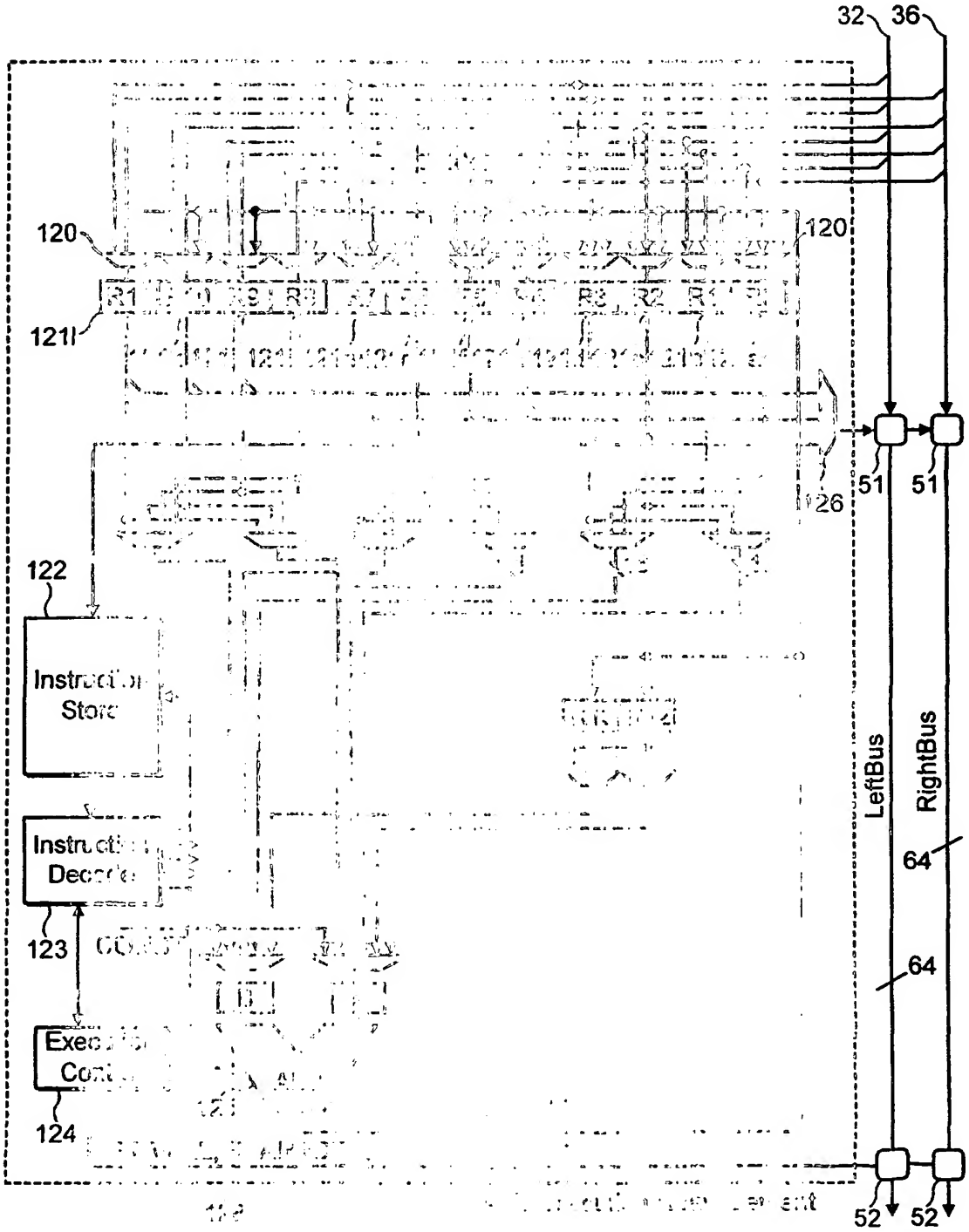
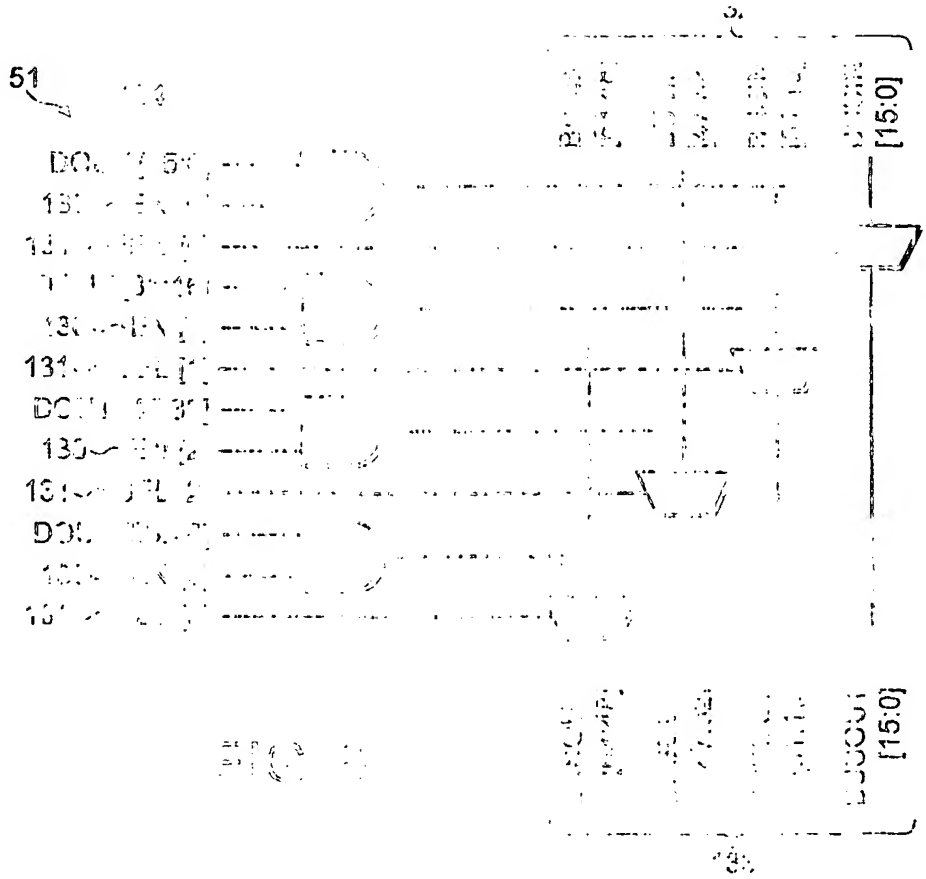
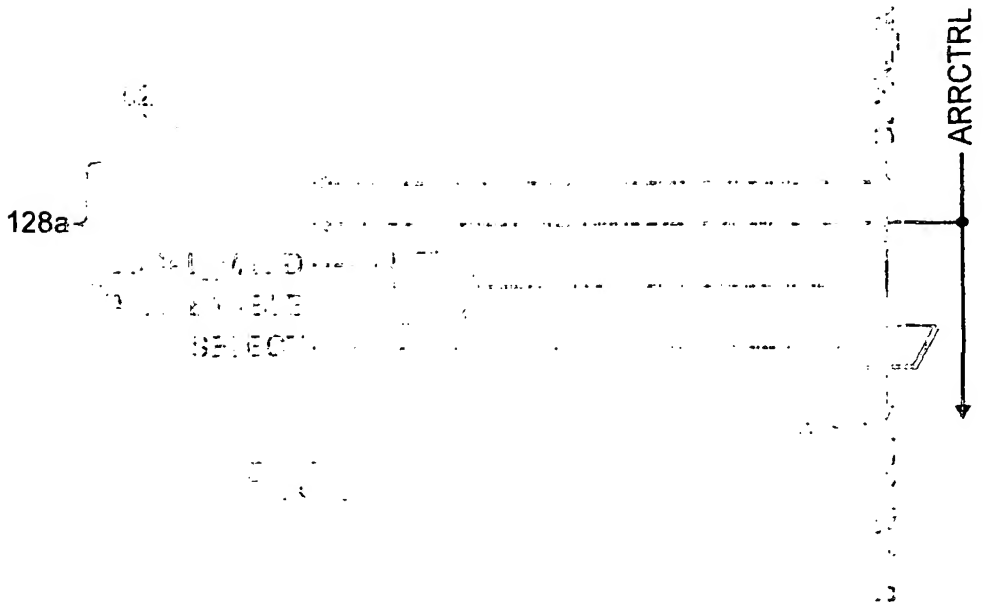


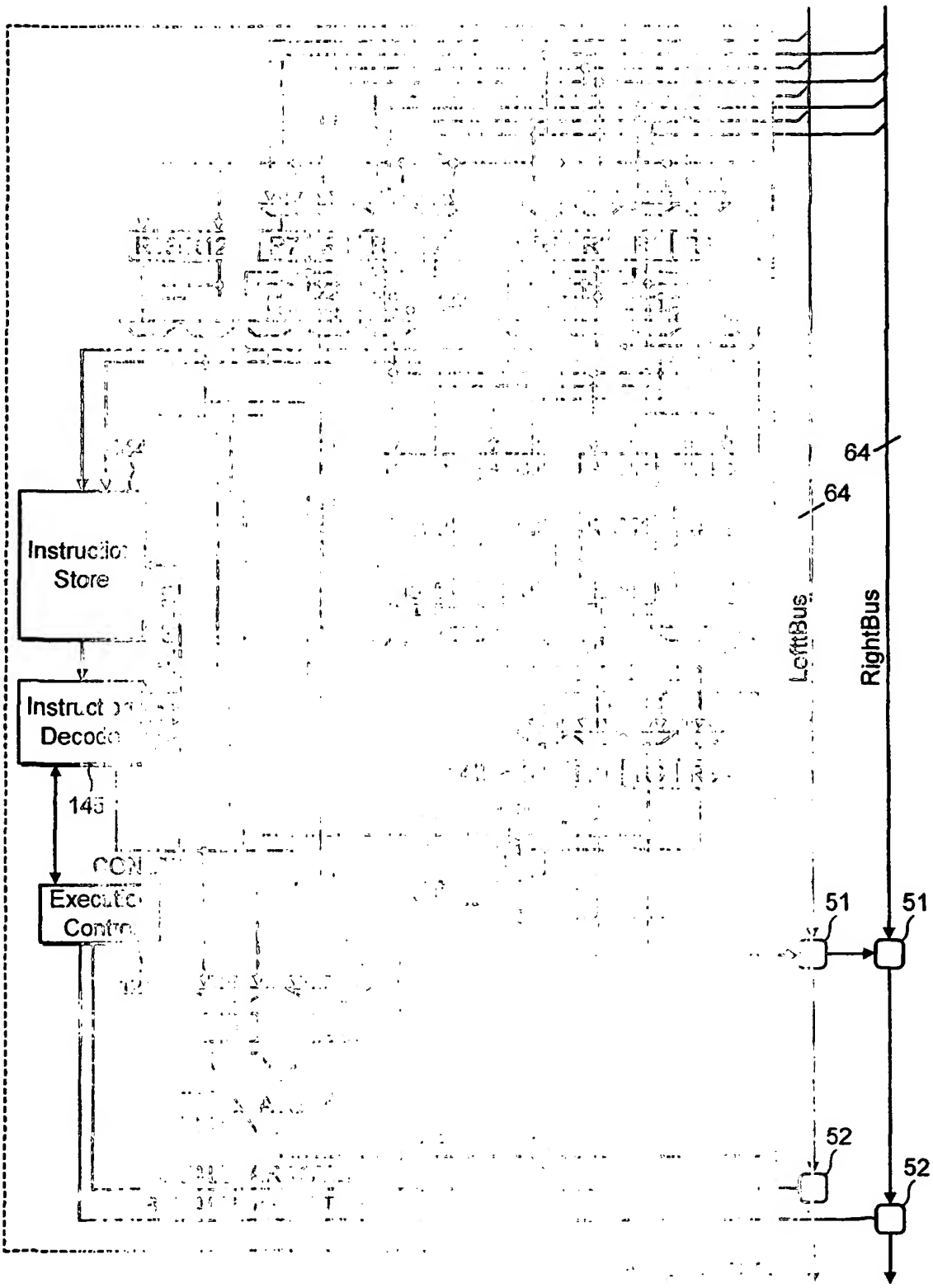
FIG. 8

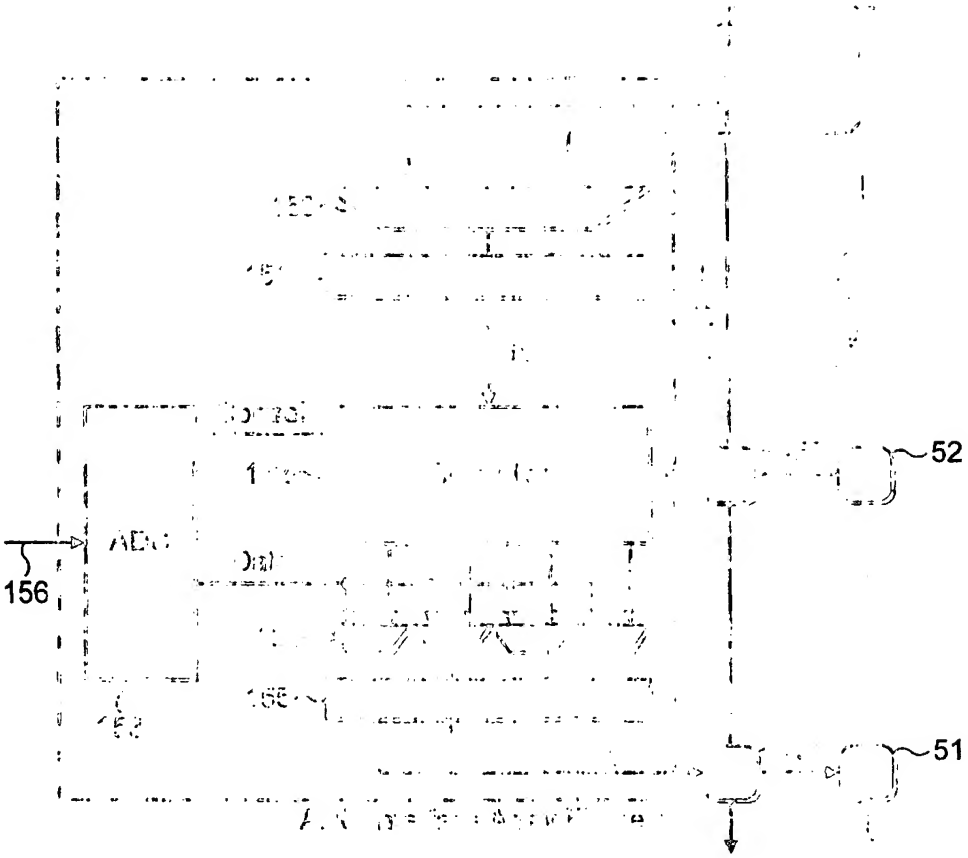

32
Emetteur
93
92
96
94
91
51
52
BD

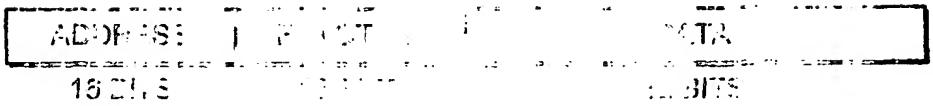

32
36
120
120
121
121
R1
R3
R3
R2
R1
26
51
51
122
Instruction Store
Instruction Decode
123
Execution Control
124
LeftBus
RightBus
64
64
52
52


51
[15:0]
[15:0]


ARRCTRL
128a


64
64
Instruction Store
LeftBus
RightBus
145
51
51
52
52


52
156
51


ADDRESS
DATA

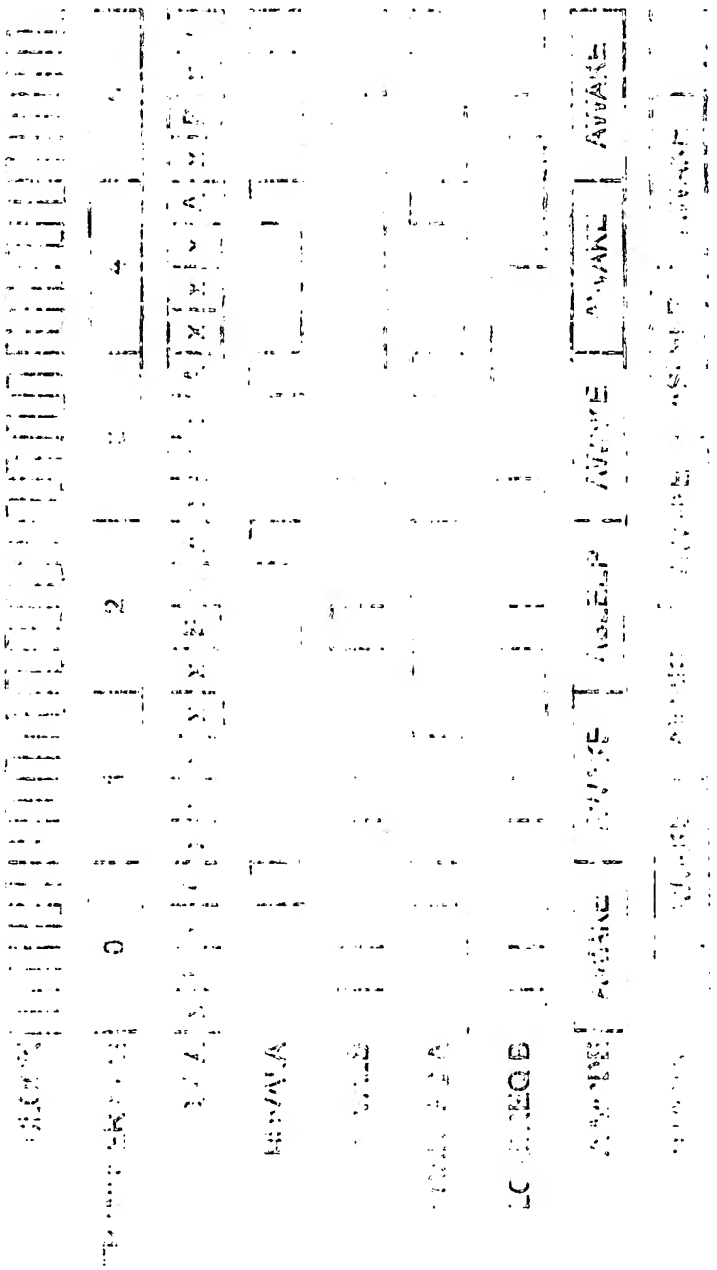


Fig. 10

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 June 2002 (27.06.2002)

(10) International Publication Number
WO 02/050700 A3

(51) International Patent Classification[7]: G06F 15/80

(21) International Application Number: PCT/GB01/05535

(22) International Filing Date:

(25) Filing Language:

(26) Publication Language:

(30) Priority Data:
0030094.8

(71) Applicant (for all designated States except US): OCHIP [illegible]
Gardens Fa[illegible]

(72) Inventors:

(75) Inventors/Applicants (for US only): [illegible]
Pete[illegible]
(GB).

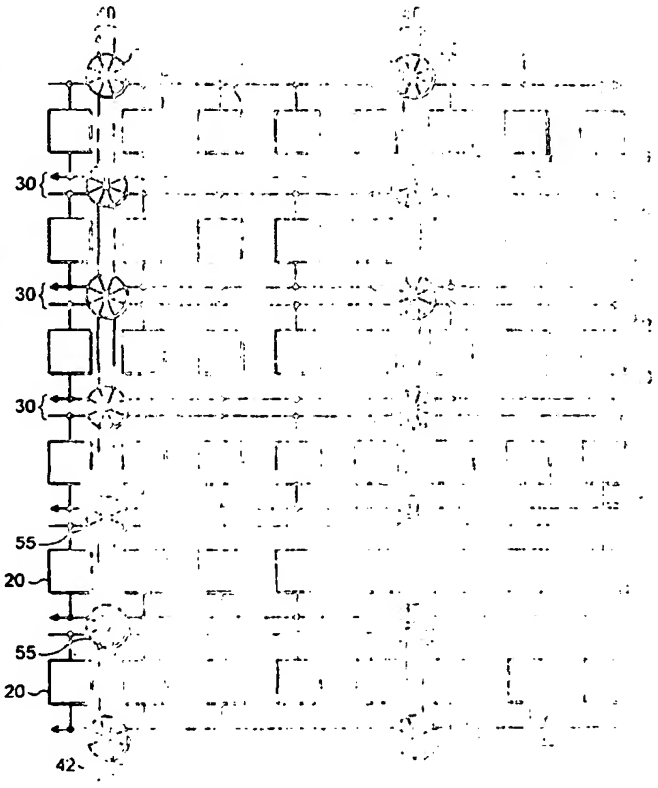
[illegible] Christopher; Haseltine [illegible] 5-19 Kingsway, London [illegible]

[illegible] AG, AL, AM, AT, AU, [illegible] CH, CN, CO, CR, CU, [illegible] ES, FI, GB, GD, GE, GH, [illegible] KE, KG, KP, KR, KZ, LC, [illegible] MD, MG, MK, MN, MW, [illegible] PT, RO, RU, SD, SE, SG, SI, [illegible] UG, US, UZ, VN, YU, [illegible]

[illegible] ARIPO patent (GH, GM, [illegible] TZ, UG, ZW), Eurasian [illegible] RU, TJ, TM), European [illegible] ES, FI, FR, GB, GR, IE, [illegible] API patent (BF, BJ, CF, [illegible] ML, MR, NE, SN, TD,

[Continued on next page]

(54) Title: PROC[illegible] ARCHITE[illegible]



[illegible] is described a processor [illegible] a plurality of processing el[illegible] having at least one input [illegible] input port, each port hav[illegible] a valid data signal line; [illegible] which contains a plurality [illegible] arranged so as to allow [illegible] first processing element [illegible] input port of any second [illegible] a time interval, in which [illegible] is enabled to set a value [illegible] data signal line of its output port to [illegible] whether the associated data bus [illegible], and to a second logic [illegible] does not contain a trans[illegible] each processing element [illegible] a waiting state for a [illegible] interval when the value on [illegible] of the associated input [illegible] logic state. This reduces [illegible] of the device.

WO 02/050700 A3

Published:

with inter[illegible]

[illegible] refer to the "Guid-[illegible] aring at the begin-[illegible]

(88) Date of p[illegible]

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G03F1 [illegible] G03F7/0

According to International Patent Classification (IPC) [illegible]

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G03

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	SCHMITT U ET AL: [illegible]	3,4, [illegible]3
A	page 23, left-hand column [illegible] GB, left-hand column [illegible] 4)	5-8

[X] Further documents are listed in the continuation of box C. [ ] See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

6 [illegible] 2

Date of mailing of the international search report

[illegible]

Name and mailing address of the ISA

[illegible]

Form PCT/ISA/210 (second sheet) [illegible]

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SCHMIDT ET AL.: "DATA-DRIVEN GRAM PROCESSING FOR ... PROCESSING OF THE XXTH ... COMPUTER ... (ICCD '90), 11 ... 8 - 12, 1990, NEW YORK, IEEE, US, vol. ... pages ... (1990-...) the whole document	1, 13
A	EP 0 473 094 A (...) 19 January 2000 (2000-01-19) page 17, line 24 - line 37; figure 1	1

Form PCT/ISA/210 (continuation ... 1992)

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0[illegible]	[illegible]	WO 9608778 A1	21-03-1996
		EP 0978339 A2	09-02-2000
		DE 69424304 T2	03-08-2000
		DE 69424304 T2	30-11-2000
		EP 0728337 A1	28-08-1996
		JP 9505981 T	03-06-1997

Form PCT/ISA/210 (patent family annex) (July 1998)